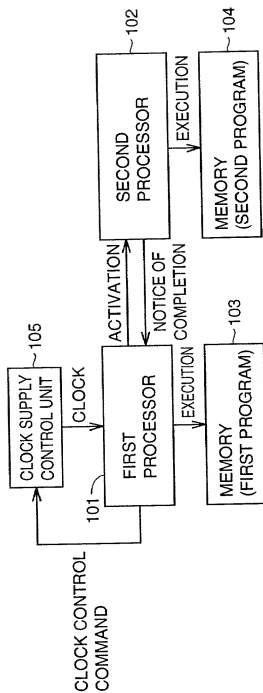


FIG. 1 PRIOR ART



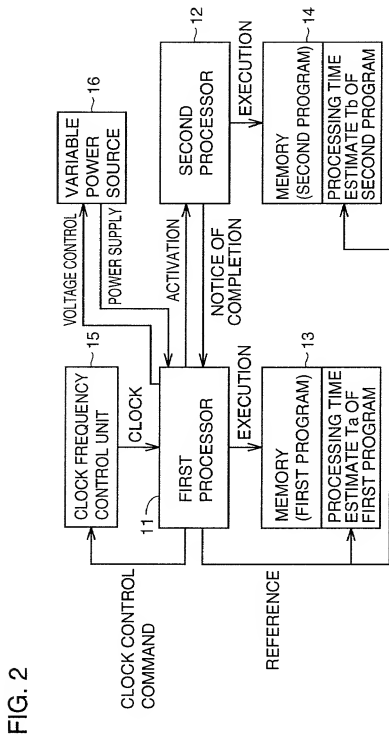


FIG. 3

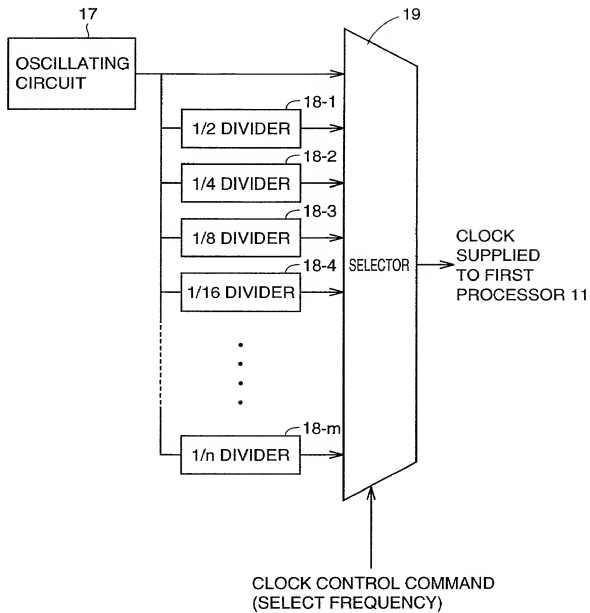


FIG. 4

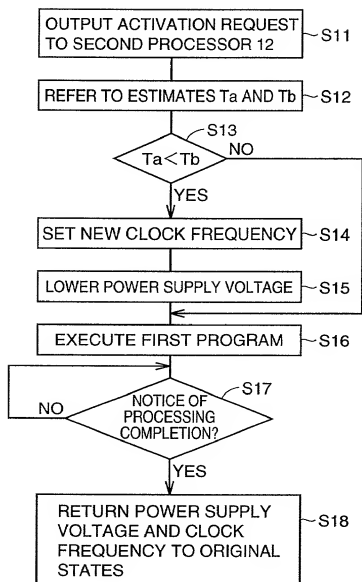


FIG. 5

```

main_procedure()
{
    B_is_done=0;
    invokeB();

    set_clock_frequency(Ta/Tb);
    set_power(table);
    do_something();

    while (B_is_done)
        /* do nothing */;
}

interrupt_from_B()
{
    restore_power();
    restore_clock_frequency();
    B_is_done=1;
}

```

/* MAIN PROCESSING UNIT OF FIRST PROGRAM */

/* ACTIVATE SECOND PROGRAM. PROCESSING TIME ESTIMATE IS Tb */

/* LOWER CLOCK FREQUENCY WITHIN LIMIT DETERMINED BY Ta/Tb */

/* LOWER VOLTAGE REFERRING TO TABLE */

/* PROCESSING TO BE DONE WITH FIRST PROGRAM. PROCESSING TIME ESTIMATE IS Ta */

/* WAIT FOR COMPLETION OF SECOND PROGRAM */

/* INTERRUPT HANDLING ROUTINE OF FIRST PROGRAM */

/* RETURN VOLTAGE TO ORIGINAL STATE */

/* RETURN CLOCK FREQUENCY TO ORIGINAL STATE */

FIG. 6

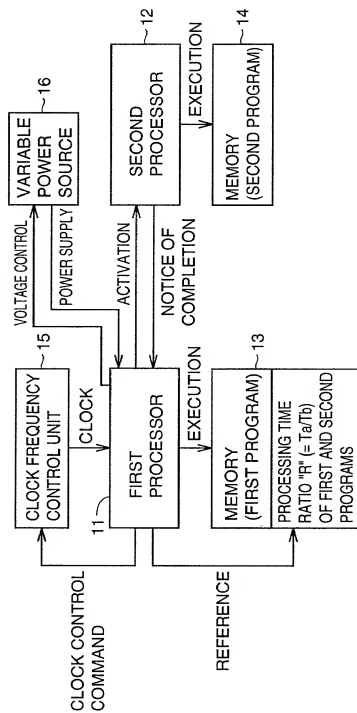


FIG. 7

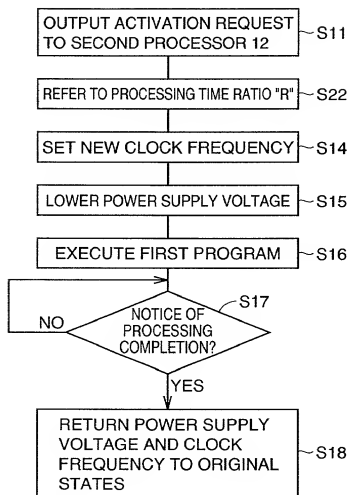


FIG. 8

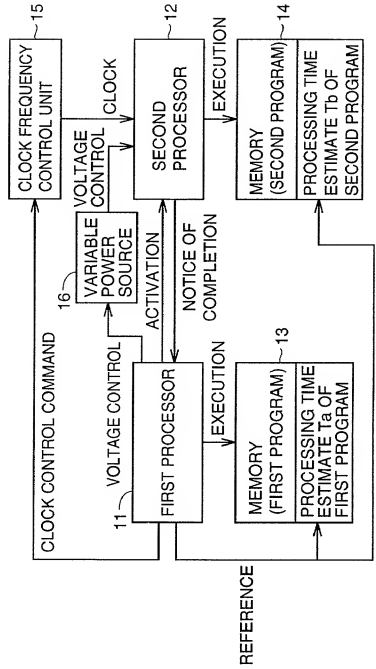


FIG. 9

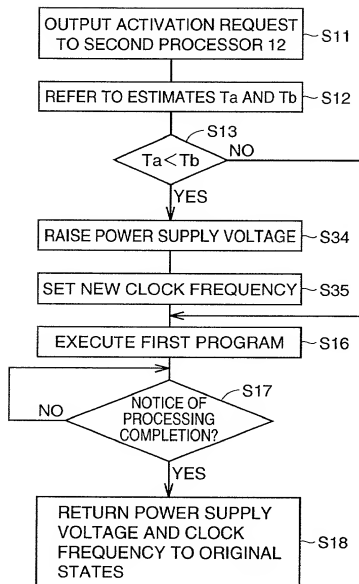


FIG. 10

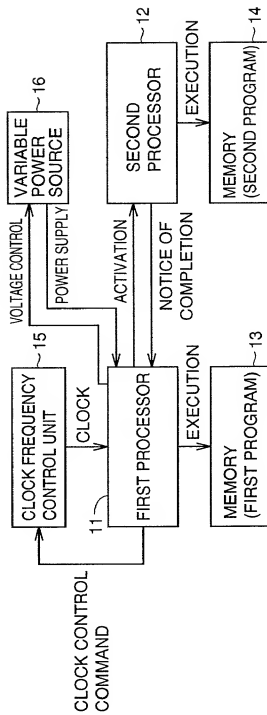


FIG. 11

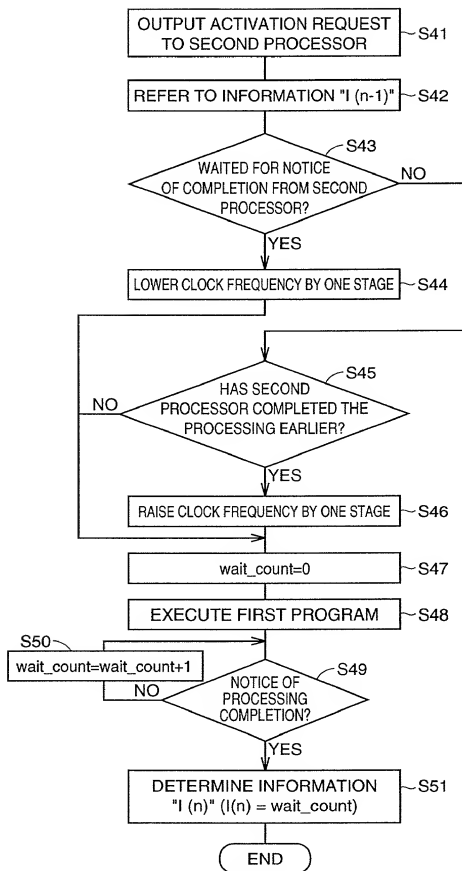


FIG. 12

```

main_procedure()
{
    B_is_done=0;
    invokeB();

    if(wait_count>WAIT_LIMIT)
        set_clock_faster();
    else if(wait_count==0)
        set_clock_slower();
    else
        /* do nothing */;
    wait_count=0;

    do_something();

    while (B_is_done==0)
        wait_count++;
}

interrupt_from_B0
{
    B_is_done=1;
}

```

/* MAIN PROCESSING UNIT OF FIRST PROGRAM */
 /* ACTIVATE SECOND PROGRAM */
 /* IF COMPLETION OF SECOND PROGRAM WAS WAITED IN PREVIOUS PROCESSING, */
 /* LOWER CLOCK FREQUENCY BY ONE STAGE */
 /* IF SECOND PROGRAM WAS COMPLETED EARLIER, */
 /* RAISE CLOCK FREQUENCY BY ONE STAGE */
 /* IF FIRST AND SECOND PROGRAMS WERE COMPLETED APPROXIMATE AT THE SAME TIME, */
 /* DO NOTHING AND KEEP CURRENT CLOCK FREQUENCY */
 /* PROCESSING TO BE DONE IN FIRST PROGRAM */
 /* WAIT FOR COMPLETION OF SECOND PROGRAM */
 /* INTERRUPT HANDLING ROUTINE OF FIRST PROGRAM */

FIG. 13

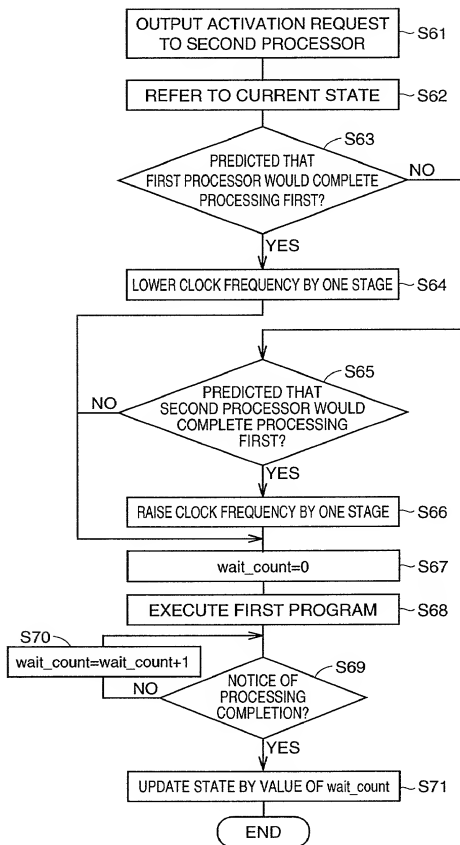


FIG. 14

```

main_procedure()
{
    B_is_done=0;
    invokeB();

    PREDICTED VALUE = OBTAIN PREDICTED VALUE FROM CURRENT STATE;
    if (PREDICTED VALUE = "PREDICT THAT PROCESSING OF FIRST PROGRAM WILL BE COMPLETED FIRST")
        set_clock_faster();
    else (PREDICTED VALUE = "PREDICT THAT PROCESSING OF SECOND PROGRAM WILL BE COMPLETED FIRST")
        set_clock_slower();
    else /* PREDICT THAT FIRST AND SECOND PROGRAMS WILL BE COMPLETED APPROXIMATELY AT THE SAME TIME */
        /* do nothing */;
    wait_count=0;

    do_something();

    while (B_is_done==0)
        wait_count++;
    UPDATE STATE BY VALUE OF wait_count;
}

interrupt_from_B()
{
    B_is_done=1;
}
    
```

/* MAIN PROCESSING UNIT OF FIRST PROGRAM */

/* ACTIVATE SECOND PROGRAM */

/* PROCESSING TO BE DONE IN FIRST PROGRAM */

/* WAIT FOR COMPLETION OF SECOND PROGRAM */

/* INTERRUPT HANDLING ROUTINE OF FIRST PROGRAM */

FIG. 15

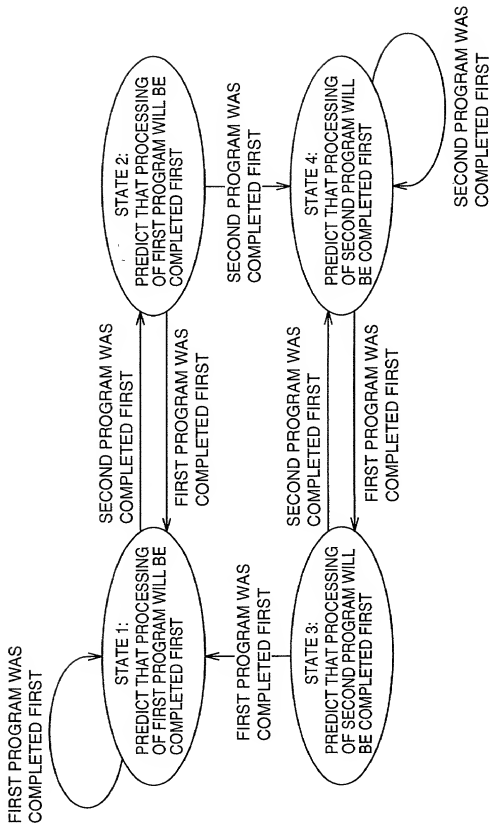


FIG. 16

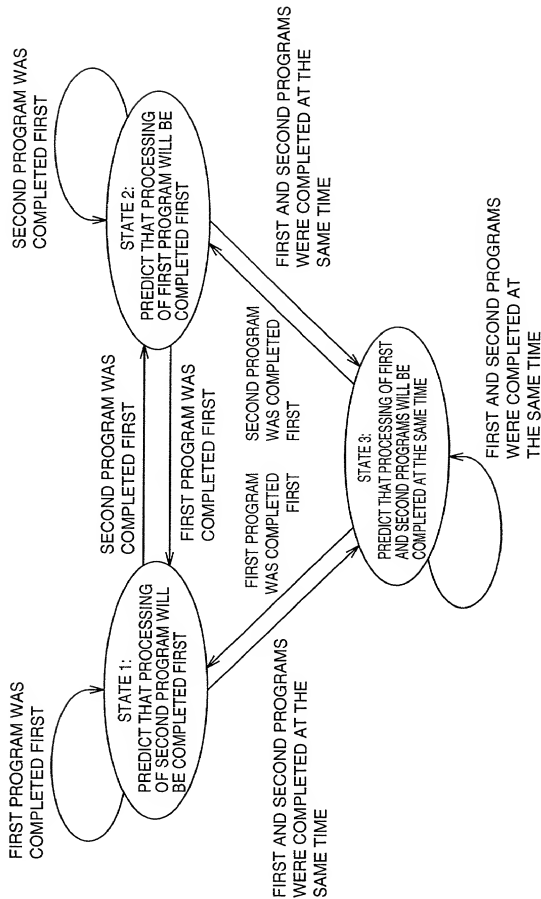


FIG. 17

